Capítulo 15

Una Experiencia (Relativamente) Insatisfactoria de Uso de Scratch en CS1

José-Alfredo Martínez-Valdés y J. Ángel Velázquez-Iturbide, *Senior Member, IEEE*

***Title*— A (Relatively) Unsatisfactory Experience of Use of Scratch in CS1.**

***Abstract*— Scratch is a "rich-media programming language" that has become very popular at high school because students may very quickly learn it and produce surprisingly animated programs. Consequently, some instructors have proposed using Scratch in introductory programming courses at the university. Their experiences report on students' high motivation and sometimes also on higher performance. We adopted Scratch as the introductory programming language for a CS1 course in a videogames major. It was used for two weeks and then the course switched to using Java. The results we obtained for both the Scratch language and the Dr. Scratch tool were less satisfactory than expected and, in some regards, disappointing. We describe our experience, analyze students' acceptance and discuss some lessons learnt to use Scratch in university courses.**

***Keywords*—Computer science education; human factors.**

***Resumen*— Scratch es un "lenguaje de programación multimedia" que se ha hecho muy popular en los institutos porque muy rápidamente los alumnos lo aprenden y producen espectaculares programas animados. En consecuencia, algunos profesores han propuesto usar Scratch en las asignaturas de introducción a la programación de la universidad. Sus experiencias han mostrado alta motivación de los alumnos y a veces también mayor rendimiento académico. En este contexto, hemos adoptado Scratch como el lenguaje de introducción a la programación en un grado de videojuegos. Se utilizó durante dos semanas y posteriormente se cambió a Java. Los resultados obtenidos tanto para el lenguaje Scratch como para la herramienta Dr. Scratch fueron menos satisfactorios de lo esperado y, en algunos aspectos, desilusionantes. En este capítulo describimos nuestra experiencia, analizamos su aceptación por los alumnos y presentamos algunas lecciones aprendidas sobre el uso de Scratch en la universidad.**

***Palabras clave*— enseñanza de la programación; factores humanos.**

I. Introducción

SCRATCH es un "lenguaje de programación multimedia" [1] que se ha hecho muy popular en los institutos, sobre todo como medio de desarrollar el pensamiento computacional [2][3]. Scratch presenta varias características que facilitan su aprendizaje. Primero, no es necesario aprender los detalles de la sintaxis de un lenguaje de programación, sino que sólo hay que recordar los nombres de las instrucciones (a veces, incluso basta con conocer sólo el efecto que producen) y seleccionarlas. Segundo, la naturaleza multimedia de los proyectos Scratch exige a los alumnos menos esfuerzo de abstracción que los tradicionales problemas numéricos o de procesamiento de información. Tercero, la inmediata realimentación visual y auditiva facilita las tareas de crear y depurar programas y aumenta la motivación de los alumnos. Cuarto, Scratch es más flexible y potente que otros esfuerzos similares del pasado, como Logo [4]. A partir de una metáfora de Seymour Papert, se ha dicho que Scratch tiene "suelo bajo, techo alto y paredes amplias" [1], es decir, es un lenguaje con el que resulta fácil iniciarse, crear proyectos de complejidad creciente en el tiempo y crear diferentes clases de proyectos. En consecuencia, los alumnos noveles pueden producir programas llamativos y complejos tras sólo unos pocos días de experiencia de programación. Finalmente, Scratch proporciona la oportunidad de colaborar y compartir a través de una comunidad web.

Estas características hacen de Scratch un lenguaje de programación también atractivo en la universidad [5]. Las investigaciones sobre aprendizaje de la programación en la universidad han permitido identificar diversas dificultades y formas de afrontarlas [6]. Los lenguajes basados en bloques [7], como Scratch, ofrecen la posibilidad de reducir algunas de estas dificultades. Actualmente existen varias experiencias

Este trabajo se presentó originalmente en la *5th International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'07)*.

Este trabajo se ha financiado con los proyectos de investigación TIN2015-66731-C2-1-R del Ministerio de Economía y Competitividad y S2013/ICE-2715 de la Comunidad Autónoma de Madrid.

José-Alfredo Martínez-Valdés trabaja en la Institución Educativa Sagrada Familia, Palmira, Valle del Cauca, Colombia (phone +573017690703; +5723069759; e-mail: ja.martinezv@alumnos.urjc.es).

J. Ángel Velázquez-Iturbide es profesor del Departamento de Informática y Estadística, Universidad Rey Juan Carlos, 28933 Móstoles, Madrid, España (phone: +34 91 664 74 54; fax: +34 91 488 85 30; e-mail: angel.velazquez@ urjc.es).

Alfonso Lago Ferreiro, André Fidalgo, Oliveira da Silva
TICAI 2017 TICs para el Aprendizaje de la Ingeniería. ISBN 978-84-8158-774-6

de uso de Scratch en asignaturas introductorias a la programación. Malan y Leitner [8] describen el uso de Scratch en un curso cero de verano sobre informática (CS0 en la terminología anglosajona). Dedicaron dos clases y dos prácticas a Scratch. Los alumnos se implicaron en el desarrollo de sus proyectos y les dedicaron más horas de las esperadas. Además, la mayoría de los alumnos sin experiencia previa en programación consideraron que Scratch les había ayudado a aprender a programar. Rizvi *et al.* [9] describen un curso CS0 de un semestre de duración ofertado para mejorar la permanencia, rendimiento y actitudes de los alumnos en riesgo de abandonar sus estudios. Una evaluación indicó que el curso ayudó a los alumnos con respecto a su autoeficacia percibida y su rendimiento.

Mishra y sus colegas [10] presentan una experiencia similar en una asignatura de introducción a la programación (CS1). Las dos primeras semanas se empezó con Scratch y luego se cambió a C++. Al igual que en la experiencia anterior, estos autores afirman que los alumnos estuvieron implicados y satisfechos. La experiencia fue satisfactoria tanto para los alumnos noveles como para los experimentados: los primeros aprendieron con poco esfuerzo y los segundos estuvieron motivados para abordar retos más complejos.

Nuestra motivación para usar Scratch en una asignatura de CS1 en un grado de videojuegos fue parecida a las experiencias anteriores. Sabíamos que los alumnos tenían distintos perfiles y pensamos que les motivaría usar Scratch al comienzo de la asignatura. En concreto, teníamos esta expectativa con los alumnos de nuevo ingreso sin experiencia de programación (que suelen tener un perfil "artístico"). Se usó Scratch durante dos semanas y después se cambió a Java. Sin embargo, los resultados que obtuvimos tanto para el lenguaje Scratch como para la herramienta Dr. Scratch no fueron tan satisfactorios como esperábamos. En el artículo, presentamos nuestra experiencia, analizamos su aceptación por los alumnos y comentamos algunas lecciones aprendidas sobre el uso de Scratch en asignaturas universitarias.

La estructura del capítulo es la siguiente. En la sección segunda, describimos la planificación de la investigación, incluyendo la asignatura, las preguntas de nuestra investigación y los instrumentos de recogida de datos. La sección tercera presenta los resultados de analizar los datos recogidos por distintos medios. Finalmente, las secciones 4 y 5 incluyen, respectivamente, un comentario de los resultados y nuestras conclusiones.

II. Planificación de la Investigación

En esta sección, describimos la asignatura, las preguntas de la investigación y los instrumentos usados.

A. La Asignatura

La asignatura "Programación visual" es una asignatura de introducción a la programación del Grado en Diseño y Desarrollo de Videojuegos. La asignatura tiene asignadas dos horas semanales para clases presenciales y otras dos horas para actividades en el aula informática.

El grado en videojuegos tiene algunas características que le diferencian de los grados tradicionales de informática. Se imparten algunas asignaturas que tradicionalmente han sido objeto de estudio en grados de arte o de diseño. Por tanto, junto a alumnos que responden al estereotipo de un alumno técnico, otros responden a un perfil más "artístico".

Afrontamos esta diversidad de alumnado en la asignatura comenzando con la introducción de un lenguaje de bloques y el cambio posterior a un lenguaje de programación textual, como Java. Se enseñó Scratch durante dos semanas y los alumnos debían entregar una práctica en la segunda semana. La evaluación se realizó en el primer cuatrimestre del curso académico 2016-17, aunque la asignatura ya había tenido esta organización en los dos cursos anteriores.

Para la práctica con Scratch, los alumnos eran libres de elegir el tema, pero tenían que utilizar varios elementos del lenguaje, entre ellos bucles, eventos, operadores y variables. La práctica se evaluó teniendo en cuenta el uso de elementos de Scratch y la funcionalidad e interfaz del proyecto.

Las prácticas también fueron evaluadas con la herramienta Dr. Scratch [11][12]. Dr. Scratch evalúa proyectos Scratch con respecto a 7 "dimensiones": pensamiento lógico, representación de datos, interactividad del usuario, control del flujo, abstracción y descomposición de problemas, paralelismo, y sincronización [11]. No damos aquí una definición detallada de las dimensiones ya que sus nombres son bastante descriptivos para una comprensión general (sin embargo, entramos en algo más de detalle en la sección de comentarios). Un proyecto puede recibir una puntuación para cada dimensión que varía entre el nivel 0 y 3, dependiendo del nivel de sofisticación que muestra el código del proyecto. Además, Dr. Scratch puede cuantificar cuatro potenciales malos hábitos [12]: atributos inicializados incorrectamente, nombres inadecuados de personajes, código duplicado y código muerto.

B. Los Alumnos

La asignatura de "Programación visual" del grado de videojuegos tenía un total de 93 alumnos matriculados en el curso académico 2016-17. Aunque los profesores sabían que los alumnos respondían a un perfil distinto del de los alumnos matriculados en grados en informática, no se había realizado ningún estudio riguroso. Pensamos que la información de matriculación era importante en sí misma y también podía ser relevante para el análisis de otros datos recopilados. Además, pensamos que los alumnos con distintos perfiles tendrían distintas expectativas y capacidades y, por tanto, podrían tener rendimiento y opiniones diferentes. Finalmente, Scratch permite crear con poco esfuerzo animaciones llamativas, lo cual parecía ser una buena forma de motivar a alumnos interesados en videojuegos.

Pedimos a los alumnos que, en su primer día de clase, rellenaran un cuestionario sobre información personal. Algunas preguntas preguntaban por datos objetivos mientras que otras pedían una clasificación subjetiva.

El cuestionario constaba de siete preguntas sobre:

- Datos personales (nombre, género y número de identificación).
- Perfil. Les pedimos a los alumnos que se autoclasificaran en una de las siguientes categorías: técnico, artístico, ambos o indefinido.
- Conocimientos de programación. Se preguntó si tenían conocimientos de programación y, en caso afirmativo, qué elementos de lenguaje conocían (de una lista cerrada) y qué lenguajes de programación conocían.

Los resultados de este cuestionario se presentan en la tercera sección.

C. Preguntas de la Investigación e Instrumentos de Medida

Abordamos dos preguntas en la investigación:

PI1. ¿Cuál es el rendimiento de los alumnos con Dr. Scratch y su opinión sobre las puntuaciones dadas por Dr. Scratch?

PI2. ¿Aceptan los alumnos el uso de Scratch como lenguaje introductorio a Java (y a la programación en general)?

Veamos los datos recogidos y los análisis realizados para responder a estas preguntas. Para PI1, se pidió a los alumnos que rellenaran un cuestionario online. Se les pidió que copiaran todos los resultados dados por Dr. Scratch a sus proyectos. Además, se pidió a los alumnos que juzgaran cuánto les había ayudado la evaluación dada por Dr. Scratch para mejorar sus habilidades de programación. Se hizo la siguiente pregunta: "¿Piensas que (Dr. Scratch) te ha ayudado a ser consciente de los aspectos en los que debes perseverar y cómo?".

Para PI2, se preparó y envió un cuestionario al final del curso. El cuestionario constaba de 4 preguntas abiertas que pedían a los alumnos que identificaran las partes de la asignatura que incluirían o suprimirían, o a las que dedicarían más o menos tiempo.

Tal y como explicamos en la siguiente sección, algunos alumnos proponían suprimir Scratch. Para aclarar este punto, hicimos dos preguntas en junio a los alumnos que suspendieron en enero. En el cuestionario, se explicaba primero el motivo de estas preguntas y después se hacían las siguientes preguntas: "¿Estás de acuerdo en suprimir Scratch de la asignatura? ¿Por qué?".

Las preguntas de múltiples opciones se han analizado usando estadística descriptiva o inferencial, según el caso. Las preguntas abiertas se analizaron con métodos cualitativos. En concreto, se analizaron según los principios de la teoría fundamentada (*grounded theory*) [13]. Dado que las preguntas abiertas eran muy concretas, no hubo una gran variedad de respuestas y la mayoría de las categorías eran previsibles.

III. Resultados

Presentamos en esta sección los resultados de nuestra investigación. Primero presentamos las características de los alumnos matriculados y después, nuestros hallazgos sobre las dos preguntas de investigación.

A. Los Alumnos

Recogimos 87 cuestionarios rellenos del cuestionario inicial. Los resultados se resumen en la Tabla I.

Obsérvese que la mayoría de los alumnos son hombres y no repetidores. Sin embargo, existe un equilibrio entre alumnos técnicos y artísticos, y entre alumnos con y sin conocimientos de programación.

Obviamente, hay diferencias en el conocimiento de programación entre alumnos repetidores y no repetidores. Sólo había un alumno repetidor que no conocía ningún lenguaje de programación, frente a un 63% de no repetidores. El lenguaje de programación más conocido por todos los alumnos fue Java, empatado con C en el caso de los alumnos no repetidores. Sorprendentemente, sólo había un alumno que conociera Scratch, y era no repetidor.

B. Rendimiento según Dr. Scratch y su Aceptación

Se recogieron 89 prácticas. Veamos resultado de su análisis.

1) Rendimiento según Dr. Scratch.

Usamos las calificaciones de Dr. Scratch como la medida del rendimiento de los alumnos en el manejo de Scratch. Dr. Scratch califica cada dimensión de un proyecto Scratch con un valor que oscila entre 0 y 3. Dado que hay 7 dimensiones, la calificación global puede variar entre 0 y 21.

Los proyectos entregados recibieron calificaciones comprendidas entre 10 y 20 (véase la Tabla II). Si descomponemos las calificaciones totales en las siete dimensiones, obtenemos los datos más detallados mostrados en la Tabla III.

La Tabla IV da las cifras de los malos hábitos detectados por Dr. Scratch.

También indagamos si había alguna correlación entre las calificaciones y los distintos factores de la población (género, perfil, etc.) No encontramos ninguna correlación.

TABLA I
CARACTERÍSTICAS DE LOS ALUMNOS MATRICULADOS (N=87)

Factor	Valor	# alumnos	%
Género	Hombre	78	90%
	Mujer	9	10%
Perfil	Técnico	39	45%
	Artístico	42	48%
	Ambos	5	6%
	Indefinido	1	1%
Repetidor	Sí	15	17%
	No	72	83%
Conocimientos de programación	Sí	42	48%
	No	45	52%

TABLA II
CALIFICACIONES DADAS POR DR. SCRATCH (N=89)

Calificación	# alumnos	%
20	2	2%
19	1	1%
17	8	9%
16	8	9%
15	10	11%
14	18	20%
13	15	17%
12	14	16%
11	7	8%
10	6	7%

TABLA III
CALIFICACIONES DADAS POR SCRATCH, DESGLOSADAS EN DIMENSIONES (N=89)

Medida estadística	Dimensiones del pensamiento computacional*							Total
	PA	PL	FC	IN	RI	AB	SN	
Media	2'18	1'90	2'28	2'01	2'01	1'17	2'21	13'76
Desviación típica	0'90	0'98	0'45	0'11	0'18	0'55	0'73	2'23
Mediana	3	2	2	2	2	1	2	14
Moda	3	1	2	2	2	1	2	14
Mínimo	1	0	2	2	1	1	1	10
Máximo	3	3	3	3	3	3	3	20

*PA: paralelismo, PL: pensamiento lógico, FC: flujo del control, IN: interactividad, RI: representación de la información, AB: abstracción, SN: sincronización

TABLA IV
MALOS HÁBITOS DETECTADOS POR DR. SCRATCH (N=89)

Mal hábito	# alumnos	%
Atributos inicializados incorrectamente	87	98%
Nombres inadecuados de personajes	47	53%
Código duplicado	25	28%
Código muerto	7	8%

2) Aceptación de Dr. Scratch

Junto al rendimiento, se analizó la opinión de los alumnos sobre la utilidad de Dr. Scratch para mejorar sus habilidades de codificación. La Tabla V muestra las respuestas dadas por los alumnos a las preguntas abiertas sobre si Dr. Scratch les había ayudado a codificar mejor, y cómo. La categoría "sí, pero…" agrupa respuestas que son positivas pero matizadas, normalmente con algún aspecto negativo de Dr. Scratch. Un ejemplo de este tipo de respuesta es la siguiente: "Sí, aunque algunos errores (…) son, a mi entender, irrelevantes" (Alumno 71, representado A71).

La Tabla VI las opiniones de los alumnos sobre la manera en que Dr. Scratch les ayudó.

La Tabla VII muestra las opiniones negativas recogidas sobre Dr. Scratch.

Veamos con más detalle las tres primeras categorías:

- El alumno no está de acuerdo con los criterios de Dr. Scratch. Un alumno no está de acuerdo con tener que incluir en su programa elementos adicionales del lenguaje para aumentar la puntuación dada por Dr. Scratch cuando la funcionalidad del proyecto ya estaba satisfecha: "(…) y exige código, que en tu proyecto puede ser innecesario, para darte mayor puntuación." (A72). Otros alumnos no están de acuerdo con que se les obligue a cambiar el nombre dado por Scratch a personajes o variables. Finalmente, algunos alumnos no están de acuerdo con otros criterios de evaluación.
- Dr. Scratch no evalúa la calidad global del programa. La mayor parte de los alumnos incluidos en esta categoría consideran que Dr. Scratch también debería evaluar el proyecto completo, no sólo aspectos parciales. Por ejemplo: "No exactamente. Mi juego lo jugué y lo di a probar a mis compañeros y pareció divertido, que es de lo que se trata un juego. Utilicé casi todas las funciones practicadas en clase, demostrando así los conocimientos adquiridos." (A89).
- Funcionalidades de Dr. Scratch. Varios alumnos se quejaron de la difícil comprensión de los mensajes emitidos por Dr. Scratch. Otro alumno opinó que Dr. Scratch no evalúa coherentemente con sus propios criterios.

TABLA V
OPINIONES SOBRE LA UTILIDAD DE DR. SCRATCH (N=89)

Respuesta	# alumnos	%
En blanco	39	44%
Sí	27	30%
Sí, pero…	9	10%
No	14	16%

TABLA VI
OPINIONES POSITIVAS SOBRE DR. SCRATCH

Respuesta	# alumnos
Ayuda a programar mejor	11
Ayuda a usar mejor algún elemento de Scratch	8
Ayuda a conocer mejor Scratch	3

TABLA VII
OPINIONES NEGATIVAS SOBRE DR. SCRATCH

Respuesta	# alumnos
No está de acuerdo con los criterios de Dr. Scratch	8
Dr. Scratch no evalúa la calidad global del programa	5
Funcionalidades de Dr. Scratch	4
Dr. Scratch no guía para programar mejor	1

C. Aceptación de Scratch como Lenguaje de Introducción a la Programación

Presentamos los resultados de los dos cuestionarios elaborados para recoger las opiniones de los alumnos sobre el uso de Scratch como primer lenguaje antes de enseñarles Java.

1) Primer Cuestionario Final

Se recogieron 76 respuestas al cuestionario presentado en el examen final de enero. Las respuestas a las cuatro preguntas abiertas del cuestionario contenían diversas opiniones y sugerencias sobre la asignatura. Sin embargo, no conviene analizar directamente las respuestas por varias razones. Primero, algunas respuestas son más adecuadas para otras preguntas. Además, algunos alumnos dieron la misma opinión varias veces, en diferentes preguntas. Por último, algunas respuestas a una pregunta son compuestas, es decir, contienen varias opiniones sencillas.

Dada esta situación, realizamos dos operaciones sobre las respuestas para su análisis posterior: agrupamos las respuestas en dos grandes categorías (positivas y negativas) y descompusimos las respuestas de cada alumno en comentarios sencillos (eliminando los duplicados).

Hubo 7 comentarios (relativamente) positivos sobre el uso de Scratch en la asignatura. Pueden agruparse en dos categorías:

- Scratch es potencialmente útil como una introducción a Java, pero debería expresarse explícitamente la relación entre ambos lenguajes (5 respuestas).
- Scratch es útil como una introducción a la programación (2 respuestas).

Sin embargo, el número de opiniones negativas fue mucho mayor. Cuarenta y un alumnos (55% del total de respuestas) pensaban que Scratch debería eliminarse o se le debería dedicar menos tiempo en el temario. Estas respuestas negativas pueden agrupare en dos categorías:

- Debería eliminarse Scratch (17 alumnos, 23%).
- Debería reducirse el tiempo dedicado a Scratch (24 alumnos, 32%). Algunas respuestas concretan el tiempo que deberían dedicarse a Scratch, variando desde una sesión a dos semanas. Obsérvese que esta última sugerencia coincide con el tiempo que realmente se dedicó en este curso académico. Por tanto, parce conveniente tomar estas opiniones como la expresión de un sentimiento más que como una sugerencia meditada. Algunas sugerencias más específicas fueron las siguientes:
  - Cambiar parte del contenido de las clases de Scratch, cambiando el tipo de ejemplos o explicando mejor su relación con Java (2 alumnos).
  - Eliminar o cambiar el carácter de la práctica de obligatoria a voluntaria (2 alumnos).

2) Segundo Cuestionario Final

En el segundo examen final (en junio), queríamos obtener una confirmación de la opinión negativa de los alumnos sobre el uso de Scratch como una introducción a la programación y a Java. Les preguntamos explícitamente si estaban de acuerdo con suprimir Scratch de la asignatura.

Se recogieron 26 respuestas, como muestra la Tabla VIII.

Analizamos en detalle las razones dadas por los alumnos.

Primero dividimos las explicaciones en explicaciones sencillas, ya que algunos alumnos dieron varias razones. De esta forma, los 9 alumnos que estaban a favor de eliminar Scratch aportaron 14 razones sencillas. Las razones resultantes pueden clasificarse en tres categorías, como muestra la Tabla IX.

De la misma forma, analizamos las razones dadas por los alumnos que no estaban de acuerdo en eliminar Scratch. Obtuvimos 19 argumentos sencillos, que pueden agruparse en tres categorías (véase la Tabla X). Obsérvese, sin embargo, que la categoría más frecuente corresponde a una opinión negativa. Fue dada por alumnos que no estaban de acuerdo con eliminar completamente Scratch, pero que pensaban que debía acortarse el tiempo dedicado. En realidad, los 8 alumnos que dieron la segunda razón también dieron la primera.

IV. Comentarios

Comentamos los resultados presentados en la sección anterior. Los resultados sobre matriculación son útiles porque confirman nuestras sospechas sobre el perfil dual de alumnos técnicos y artísticos. Para nuestra sorpresa, también muestran que Scratch no es tan conocido por los alumnos como esperábamos. Sin embargo, las características de los alumnos no es el objeto principal de nuestro estudio. Esperábamos encontrar alguna correlación entre algunos resultados y algún perfil, pero no fue así.

Por tanto, sólo comentamos los resultados obtenidos sobre Dr. Scratch y sobre Scratch.

A. Rendimiento de los Alumnos

Las puntuaciones dadas por Dr. Scratch están en la franja media de las posibles puntuaciones, con una media igual a 13’76 y con mediana y moda iguales a 14. Estos resultados son coherentes con otros resultados obtenidos por Moreno-León *et al.* [9]. En su estudio, alumnos de instituto obtuvieron valores respectivos de 11’5, 11 y 10 en las tres medidas anteriores. Dado que sus alumnos tenían entre 10 y 14 años de edad y que nuestros alumnos eran de primer año de Universidad, era de esperar un rendimiento mayor en éstos que en aquéllos.

Los resultados obtenidos en las siete dimensiones analizadas por Dr. Scratch son más difíciles de interpretar. Los alumnos obtuvieron más puntuación en algunas dimensiones que en otras. En concreto, la puntuación obtenida en la dimensión de abstracción y descomposición de problemas (AB, 1’17) es mucho más baja que para el resto de dimensiones, cuya media es igual a 1’9 o más. El pensamiento lógico también puntúa menos que las demás dimensiones, pero las diferencias no son tan abruptas (PL, media=1’9, moda=1, mínimo=0). En el otro extremo, colocamos el paralelismo (PA, media=2’18, mediana=3), control del flujo (CF, media=2’28, mínimo=2), interactividad (IN, mínimo=2) y sincronización (SN, media=2’21).

Puede haber varias explicaciones para estos resultados dispares. Puede ser que algunas características de Scratch sean simplemente más intuitivas que otras. Por ejemplo, el paralelismo resulta natural en Scratch, al contrario que en otros lenguajes, como Java. También podría ser que algunas características sean más adecuadas que otras para implementar los proyectos más frecuentes de los alumnos (juegos, simulaciones, etc.) Por ejemplo, los clones (AB, nivel 3) son una característica avanzada y las operaciones lógicas (LT, nivel 3) no se utilizan con mucha frecuencia. Por el contrario, el bloque repetir-hasta (CF, nivel 3) o el bloque esperar-hasta (SN, nivel 3) se usan frecuentemente para implementar proyectos de complejidad no trivial. Una última explicación de los dispares resultados podría ser que el diseño de las dimensiones de Dr. Scratch y sus niveles deben revisarse. Estas cuestiones deberían abordarse en trabajos futuros.

Tabla VIII
Opiniones sobre Suprimir Scratch (N=26)

Respuesta	# alumnos	%	# alumnos que dan explicación
Sí	15	58%	9
No	11	42%	11

Tabla IX
Razones para Suprimir Scratch (N=14)

Razón	# alumnos
No ayuda a aprender Java	7
El tiempo dedicado a Scratch podría dedicarse a otro tema	5
Se estudia en el instituto	2

Tabla X
Razones para no Suprimir Scratch (N=19)

Razón	# alumnos
Consume un tiempo que podría dedicarse a otro tema	9
Ayuda a aprender a programar	8
Es entretenido	2

B. Aceptación de las Puntuaciones de Dr. Scratch

El porcentaje de alumnos que consideraron que Dr. Scratch había sido útil (40%, contando las categorías “sí” y “sí, pero…”) es mayor que el de aquellos que tenían alguna opinión negativa (26%, contando las categorías “sí, pero…” y “no”). No obstante, el porcentaje de opiniones negativas es mayor de lo que nos parece deseable en una herramienta educativa.

Los alumnos con una opinión positiva afirmaban que Dr. Scratch les ayudó a codificar mejor. Aquellos que concretan más su opinión afirman que, gracias a Dr. Scratch, eran más conscientes de algunas características del lenguaje o que incluso habían sabido de su existencia.

Sin embargo, hay varias quejas sobre Dr. Scratch. Algunos alumnos se quejan de lo críptico que son sus mensajes. Esta crítica es similar al problema encontrado en programadores noveles para comprender los mensajes de compilación [14].

Otras críticas están relacionadas con el diseño actual de Dr. Scratch. La herramienta evalúa el uso de los distintos elementos del lenguaje por parte del usuario. En algunos contextos, el enunciado de una práctica puede expresarse en términos de los elementos del lenguaje a usar [8] (en realidad, así lo hicimos). Sin embargo, es más frecuente que el enunciado de una práctica se escriba en términos de la funcionalidad que debe tener el programa. Por esta razón, los alumnos se decepcionan cuando piensan que su programa está bien desarrollado (a veces de común acuerdo con sus compañeros) y Dr. Scratch sólo les da una puntuación media.

Otra posibilidad sería dejar que el profesor pueda seleccionar qué dimensiones quiere evaluar y el nivel de maestría que espera en cada dimensión. De esta manera, Dr. Scratch podría evaluar los proyectos de forma alineada con los objetivos de aprendizaje de la práctica.

C. Percepción de Scratch como Lenguaje de Introducción a la Programación

El resultado más inesperado fue el amplio consenso existente entre los alumnos en considerar que debería dedicarse menos tiempo a Scratch, o que incluso debería eliminarse. Obviamente, no se alcanzaron nuestras expectativas de diseño de la asignatura, ya que los alumnos no resultaron motivados. Hay varias razones por las que los alumnos pedían dedicar menos tiempo. Primero, algunos alumnos asocian Scratch con la educación preuniversitaria, por lo que le consideran un lenguaje para niños y "poco serio". Un alumno utilizó justamente este calificativo en su respuesta al segundo cuestionario final: "No resulta útil para aprender la programación más «seria»" (A03). Otros alumnos afirman que, dado que el objetivo de aprender Scratch es facilitar el aprendizaje de Java, debería presentarse explícitamente la relación entre ambos lenguajes. Además, dado el poco tiempo dedicado a Scratch, debería ser voluntaria la realización de la práctica.

Algunas opiniones negativas afectan al profesor. Ya hemos visto que algunos alumnos piden explicaciones explícitas de la relación entre Scratch y Java. Además, algunos alumnos piden ejemplos más interesantes y menos rutinarios. Finalmente, los alumnos padecieron la falta de tiempo para estudiar el último tema de la asignatura. Por tanto, pensaban que si no se hubiera incluido Scratch, habría podido dedicarse más tiempo al último tema.

V. Conclusiones

Hemos introducido Scratch en las primeras semanas de una asignatura de introducción a la programación (del grado de videojuegos). Sin embargo, los alumnos aceptaron peor la experiencia que en otras experiencias documentadas. La insatisfacción no solo afecta al lenguaje Scratch sino también a la herramienta de evaluación Dr. Scratch. Hemos señalado varias razones de esta insatisfacción. Con respecto a Dr. Scratch, se recomienda revisar los criterios de evaluación, y proporcionar un esquema de evaluación más flexible (donde la puntuación esperada no sea siempre 3 en todas las dimensiones). Además, los profesores no deberían usar las calificaciones de Dr. Scratch cómo el único dato para la nota sino que, al menos, deberían evaluar la funcionalidad del proyecto.

Con respecto al lenguaje Scratch, la conclusión principal es que una introducción informal a Scratch, como la realizada en diversas experiencias, es insatisfactoria para algunos alumnos universitarios. El profesor debería explicar que Scratch no es un lenguaje de programación ni "de juguete" ni para niños. Deberían presentarse explícitamente los conceptos de programación y la relación de las características de Scratch con las del siguiente lenguaje de programación. Finalmente, también debería revisarse el diseño de ejemplos y prácticas.

Agradecimientos

Agradecemos a Raquel Hijón Neira su ayuda en la asignatura "Programación Visual".

Referencias

[1] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, e Y. Kafai, "Scratch: Programming for all,". *Communications of the ACM*, vol. 52, no. 11, pp. 60-67, 2009.

[2] F. J. García-Peñalvo, D. Reimann, M. Tuul, A. Rees, e I. Jormanainen, "An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers," TACCLE3 Consortium;: Bélgica.

[3] S. Y. Lye, y J. H. L. Koh, "Review on teaching and learning of computational thinking through programming: What is next for K-12?," *Computers in Human Behavior*, vol. 41, pp. 51-61, 2014.

[4] S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas*. Nueva York: Basic Books, 1980.

[5] U. Wolz, H. H. Leitner, D. J. Malan, y J. Maloney, "Starting with Scratch in CS 1," en *Proc. 40th SIGCSE Technical Symposium on Computer Science Education. SIGCSE'09*. ACM, Nueva Nueva York, NY, 2009, pp. 2-3.

[6] A. Robin, J. Roundtree, y N. Roundtree, "Learning and teaching programming: A review and discussion," *Computer Science Education*, vol. 13, no. 2, pp. 137-172, 2003.

[7] C. Kelleher, y R. Pausch, "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers," *ACM Computing Surveys*, vol. 37, no. 2, pp. 83-137, 2005.

[8] D. J. Malan, y H. H. Leitner, "Scratch for budding computer scientists," en *Proc. 38th SIGCSE Technical Symposium on Computer Science Education. SIGCSE'07*. ACM, Nueva York, NY, 2007, pp. 223-227.

[9] M. Rizvi, T. Humphries, D. Major, M. Jones, y H. Lauzun, "A CS0 course using Scratch," *Journal of Computing Sciences in Colleges*, vol. 26, no. 3, pp. 19-27, enero 2011.

[10] S. Mishra, S. Balan, S. Iyer, y S. Murthy, "Effect of a 2-week Scratch intervention in CS1 on learners with varying prior knowledge," en *Proc. 2014 Conference on Innovation and Technology in Computer Science Education. ITiCSE'14*. ACM, Nueva York, NY, 2014, pp. 45-50.

[11] J. Moreno-León, M. Román-González, y G. Robles, "Dr. Scratch: Automatic analysis of Scratch projects to assess and foster computational thinking," *RED. Revista de Educación a Distancia*, vol. 46, no. 10, 2015.

[12] J. Moreno-León, y G. Robles, "Automatic detection of bad programming habits in Scratch: A preliminary study," en *Proc. 2014 Frontiers in Education Conference. FIE 2014*, IEEE, 2014, pp. 1-4.

[13] B. Glaser, y A. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Chicago: Aldine, 1967.

[14] B. A. Becker, G. Glanville, R. Iwashima, C. McDonnell, K. Goslin, y C. Mooney, "Effective compiler error message enhancement for novice programming students," *Computer Science Education*, vol. 26, no. 2-3, pp. 148-175, 2016.

J. Ángel Velázquez Iturbide recibió los títulos de Licenciado y de Doctor en Informática por la Universidad Politécnica de Madrid, España, en los años 1985 y 1990, respectivamente. Actualmente es Catedrático de Universidad en la Universidad Rey Juan Carlos, donde también es el Director del Laboratorio de Tecnologías de la Información para la Educación (LITE, http://www.lite.etsii.urjc.es/). Sus áreas de investigación son *software* y metodologías docentes para la enseñanza de la programación y la visualización del *software*. Actualmente, es Presidente de la Asociación para el Desarrollo de la Informática Educativa (ADIE). Es miembro senior de IEEE (IEEE *Computer Society* e IEEE *Education Society*) y de ACM (y ACM SIGCSE).

José-Alfredo Martínez-Valdés recibió los títulos de Licenciado en Matemática y Física y de Especialista en Educación Matemática por la Universidad del Valle en 1994, Cali – Colombia; de Magíster en la Enseñanza de las Ciencias Exactas y Naturales en 2011 por la Universidad Nacional de Colombia y de Máster en Ingeniería de Sistemas de Decisión 2011 por la Universidad Rey Juan Carlos de Madrid, España. Actualmente es profesor a tiempo completo nombrado en propiedad en la Entidad Territorial Palmira, Ministerio de Educación de Colombia, donde se desempeña como profesor de Matemáticas y Tecnología e Informática. Además, es director del Grupo de Investigación Interacción Persona Ordenador en Bachillerato (IPOBA), registrado en COLCIENCIAS (http://www.colciencias.gov.co/)